Paper 101-2012

How to Measure the SAS® BI Audience and Discover Information Needs

Plinio Faria, São Paulo, Brazil

ABSTRACT

It's very useful to measure an audience using SAS® Web Report Studio. It helps you create better reports and know the users' information needs, and it will guide the development of your reporting system.

Many times, there is too much information available, and users feel that it is too difficult to get what they really need. It is important to create comprehensive reports and to keep only those that are being regularly viewed.

This presentation shows how to measure how often a report is accessed using SAS® Web Report Studio log files, and how to combine this data with user data to classify users into groups. You can analyze what type of reports should be available and the way users like to see these reports.

INTRODUCTION

Analyzing the SAS Web Report logs you may determine how often the reports are being opened or created and this data will be a very useful tool to understand what information is really important. This analysis is more efficient if most users do not create reports, i.e. there are few reports with lots of access.

It's very useful to measure the audience of SAS Web Report to create better reports and to know the users' needs of information in order to guide the development of your reporting system.

However, you have to keep in mind that each class of users has different needs according to its hierarchy level and to classify the users into groups according to its hierarchy level you'll need a list of users.

This measurement method has been used in a sample of over 30,000 accesses a month but it can be used for a much smaller sample.

IMPORTING LOG FILES

By default SAS Web Report log files are generated every 24 hours in the following path of the middle tier path: /suportedbdc/apl/sasconfig/Lev1/Web/Logs. These files are in XML format and contain basically user ID, Report name, action (logon, logoff, open or save) and date. The log rollover properties can be changed using SAS Management Console.

The programs below will import all files and consolidate them in a data set named log_webreport.

The macro import_log imports a log file and store the data in WEBREPORT_log dataset. The macro find_files search all *.log files in the given directory and calls log_import for each file.

```
libname LOGS "\\mz-fw-ap-007\T\logcic";

%macro log_import(file_name);
libname LogXML xml &file_name;

/* Create the Logs' dataset */
%if %sysfunc(exist(LOGS.WebReport_log))=0 %then
    %do;
           data LOGS.WebReport_log;
           set
           LogXML.Event;
           run;

           proc sql;create unique index acess on LOGS.WebReport_log
(javadate,"user"n,code);
    %end;
```

```
/* Search for all files in the folder and store them in the FILENAMES dataset */
filename logpath "%bquote(&path.)";

data filenames (keep= i file_name);
  handle=dopen( 'logpath' );
  if handle > 0 then do;
    count=dnum(handle);
    do i=1 to count;
      file_name=dread(handle,i);
      output filenames;
    end;
  end;
  rc=dclose(handle);
run;
filename logpath clear;

/* Import all log files, i.e. those ones that contain ".log_" in the name */
proc sql;
select max(i)
into :filecount
FROM filenames;

%do n=1 % to &filecount;
    proc sql;
    SELECT catt(trim("&path.\"),trim(file_name)) as nm_arq,
              find(file_name,".log_") as position
    into :nm_arq , :position
    FROM filenames
    WHERE i=&n;

    %if &position > 0  %then
          %log_import("&nm_arq.");
    ;
%end;

%mend;

/* Replace following path for the location of xml logs */
%file_search(\\d8041s024\Logs\xml);
```

REPORT GROUPS

Usually there are multiple versions of a report and all versions must be treated as one or even as a group of reports. For this reason it will be necessary to create an auxiliary dataset to group the reports. For example:

Report_name	Standardized Report Name	Report Group
Sales vs Target version 1	Sales vs Target	Sales
Sales vs Target version 2	Sales vs Target	Sales
Sales orders	Sales orders	Sales

IMPORTING USERS LIST

In order to classify users into groups it will be necessary to have a list of users containing ID , job title, region and other variables that may be important to analyze the accesses .
If you don't have a complete list of all users and assuming your profile contains the Metadata administrator role, you can run the following code to get a list of users :

```
data alllogins;
length uri
authdomain
Login
uri2
Identity $256
Userid
password $32
Group $32;
call missing(nobj,uri,authdomain,login,password,userid,uri2,Identity);

/* Determine how many logins are on this server for the current user. */
nobj=metadata_getnobj("omsobj:Login?@Id contains '.'",1,uri);

/* Iterate thru the logins objects and obtain the attributes. */
if (nobj >0) then do n = 1 to nobj;
nobj=metadata_getnobj("omsobj:Login?@Id contains '.'",n,uri);
*put nobj=; /* Number of Logins objects found. */
*put uri=; /* Nth Login. */
if (nobj>0) then do;;
rc= metadata_getattr(uri, "Name", Login);
rc= metadata_getattr(uri, "Userid", Userid);
rc= metadata_getattr(uri, "Group", Group);

/*Now retrieve the associated authentication domain. */
arc=metadata_getnasn(uri,"Domain",1,uri2);
if (arc >0) then
aurc = metadata_getattr(uri2, "Name", AuthDomain);

/* Also retrieve the Identity associated to the Login. */
arc=metadata_getnasn(uri,"AssociatedIdentity",1,uri2);
aurc = metadata_getattr(uri2, "Name", Identity);
put Login= authDomain= userid= Identity= Name= ;
output;
end;
end;
else put 'No Logins for user';
/* keep authdomain Identity Userid;*/;
run;
```

Here is another way to obtain the users list. Firstly, generate an XML file containing metadata information:

```
filename MTDT "X:\COMP_TEST\metadata.xml" encoding="utf-8";

proc metadata
IN='<GetMetadataObjects>
<Reposid>$METAREPOSITORY</Reposid>
<Type>Person</Type>
<Ns>SAS</Ns>
<Flags>257</Flags>
</GetMetadataObjects>'
out= MTDT;
run;
```

Before converting the XML file, it's necessary to create an XML map (use the example code below) or you can use SAS XML Mapper. There is much information on the file generated below, but the map above will retrieve Name, Login and Job Title.

```
<?xml version="1.0" encoding="utf-8"?>
<SXLEMAP xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="GroupMap"
version="1.2"
xsi:noNamespaceSchemaLocation="http://www.sas.com/xml/schema/sxle12.xsd">
<TABLE name="groups">
<TABLE-PATH
syntax="XPath">/GetMetadataObjects/Objects/Person/IdentityGroups/IdentityGroup</TAB
LE-PATH>

<COLUMN name="Name" retain="YES">
<PATH syntax="XPath">/GetMetadataObjects/Objects/Person/@Name</PATH>
<TYPE>character</TYPE>
<DATATYPE>string</DATATYPE>
<LENGTH>200</LENGTH>
</COLUMN>


<COLUMN name="JobTitle">
<PATH syntax="XPath">/GetMetadataObjects/Objects/Person/@Title</PATH>
<TYPE>character</TYPE>
<DATATYPE>string</DATATYPE>
<LENGTH>200</LENGTH>
</COLUMN>


<COLUMN name="Userid">
<PATH syntax="XPath">/GetMetadataObjects/Objects/Person/Logins/Login/@Id</PATH>
<TYPE>character</TYPE>
<DATATYPE>string</DATATYPE>
<LENGTH>200</LENGTH>
</COLUMN>

</TABLE>
</SXLEMAP>

libname users xml "\SAS\metadata.xml" xmlmap="\SAS\metadata.map";
proc sql;
create table users as
select distinct *
FROM users.Groups;
```

Note that this code brings information stored on users Metadata server and perhaps the job title may be outdated.

CLASSIFYING USERS

In a similar way as reports, users need to be classified into groups such as Department, Job Title or Region. Job Titles need a special attention especially because there are often many degrees for a job (Manager I, Manager II and so on).

ACCESS INDICATORS

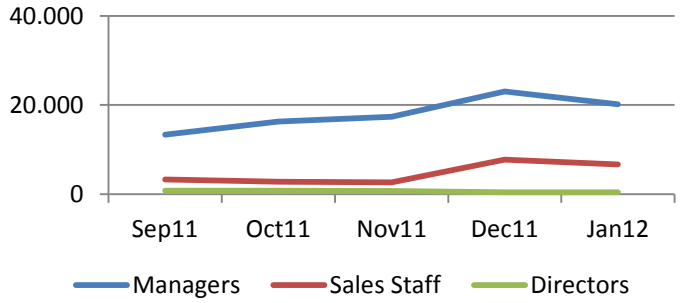
There are many possible indicators for access measurement, but the main ones are:

- Overall accesses
- Access by users class
- Acess by report

This is an example of a query that provides the number of access grouped by job title and report during a period:

```
proc sql;
select
    JobTitle,
    standard_report_name,
    count(*) as qtd format commax20.
FROM LOGS.WEBREPORT_LOG a left join report_groups b on a.report=b.report_name
                                    left join USERS c on a.USER=c.userid2
WHERE description="Open" AND "01JAN12"d <="DATE"n <= "30JAN12"d
group by 1,2;
```

You may also build an Olap cube to explore data and have insights. This is an example using SAS Olap Cube Studio and SAS Add-In.



Graph 1. Sales by job title

REPORT AUTHORS

Report authors must always check the Audience Reports. It is very satisfying to know that your work is being useful to other people.

If a report is not being opened, it must be deleted to save efforts on maintenance and also to keep the reporting system the most comprehensive as possible.

LEARN WITH USERS

Indentify heavy users and try to learn with them. Many times when users are exporting data and then regroup it, that is a clear sign that there are missing views of reports or they are not well acquainted with SAS Web Report functions.

CONCLUSION

There are lots of challenges in creating and maintaining a BI system and to be successful it is necessary to know the enterprise need of information in all levels of users. Many times it is better to have a few comprehensive reports than lots of reports plenty of detailed information that users seldom need. The same way Key Proccess Indicators (KPI) are created using SAS BI tools to analyze business process, you can use KPIs to analyze your reporting system.

REFERENCES

SAS® 9.2 Language Interfaces to Metadata. SAS Institute Inc. 2009. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Plinio Faria
Rua Luis Coelho, 53 apto 1A
Sao Paulo-SP
Brazil
Phone: + 55 11 9808-3632
E-mail: plinio.plinio@gmail.com